Generate sequential patterns

Nova F. Smedley

2018-11-04

The package `gbmSPM` can be used to do the exact analysis pipeline used in the paper,

Smedley, Nova F., Benjamin M. Ellingson, Timothy F. Cloughesy, and William Hsu. "Longitudinal Patterns in Clinical and Imaging Measurements Predict Residual Survival in Glioblastoma Patients." *Scientific reports* 8, no. 1 (2018): 14429.

See also Supplemental Materials.

Vignette Info

This vignette shows how temporal patterns can be found with sequential pattern mining (SPM) using dummy patient data and depends on the `arulesSequences` R package.

Example

1. **Set the `arulesSequences` parameters**

   - `minSupp` min. percentage of patients, i.e., sequences, with pattern
   - `maxgap` max. time between CONSECUTIVE elements in sequence, i.e., max. days between visits
   - `maxlen` max. length of elements in sequence
   - `maxsize` max. number of items in an element of a sequence

   E.g., if min. support is 0.4, only patterns that occur in 40% of patients will be returned.

   E.g., if max. length is 2, max. gap is 60, patterns are created where there is a sequence of up to 2 visits, with up to 60 days between consecutive visits

   E.g., if max. size is 2, there can be up to two events included in a single visit.

   ```r
   library(gbmSpm)

   minSupp <- 0.4
   maxgap <- 60
   maxlen <- 2
   maxsize <- 2
   ```

2. **Set the gbmSPM parameters**

   - `tType` the type of tumor volume measurment to use in generating patterns
   - `chemoOverlap` whether to only use events during the course of chemotherapy (only really relevant for descriptive statistics)
   - `suppList` can subset other sets patterns based on level of min. support specified in this list (reduces duplicated work)

   E.g., if `tType` is 'rate', rate changes will be calculated from tumor volumes and used

   ```r
   tType <- 'rate'
   chemoOverlap <- F
   suppList <- seq(minSupp, 0.4, .05)
   ```

3. **Setup experiment to save generated patterns and metadata**

   We wil put the created features in '~/gbm_spm_example', but this can be changed.

```
outputDir <- '~/gbm_spm_example'
spmDir <- file.path(outputDir,'spm')
logDir <- file.path(outputDir,'spm_logs')
lapply(c(outputDir, spmDir, logDir), function(i) ifelse(!dir.exists(i), dir.create(i, showWarnings
```

   Log experiment:

```
# if you want to save the log file, uncomment
# logfn <- file(file.path(logDir,paste0(paste0('g',maxgap,'l',maxlen,'z',maxsize),'_',
#                                         getVolTypeName(tType),'.log')), open='wt')
#
# sink(logfn, type='output', split=T)

cat(format(Sys.Date(), format="%Y.%m.%d"), '\n')
#> 2018.11.04
cat('maxgap: ', maxgap, '\n')
#> maxgap:  60
cat('maxlen: ', maxlen, '\n')
#> maxlen:  2
cat('maxsize: ', maxsize, '\n')
#> maxsize:  2
cat('tumor vol variable type: ', getVolTypeName(tType), '\n\n')
#> tumor vol variable type:  rateChange
```

4. **Load dummy data**

```
data("fake_data")
names(fake_data)
#> [1] "person" "demo"   "events"
colnames(fake_data$person)
#> [1] "iois"     "survival" "status"
colnames(fake_data$demo)
#> [1] "iois"      "DOB"       "gender"    "ethnicity" "MSP"       "IDH1"
colnames(fake_data$events)
#> [1] "iois"         "agent"        "agent_detail" "startdate"
#> [5] "enddate"

# save tumor location and laterility strings before event cleaning
fake_tumorInfo <- fake_data$events
fake_demo <- fake_data$demo

fake_data$events <- cleanData(fake_data$events, tType)
#>
#>
#>  Cleaning data...
#>
#> ... 33  events were removed due to empty agent_details
#> ... 3  events were removed due to agent_detail = "-"
#> ... 0  events were removed due to mental status = 9
#> ... 0  events were removed due to overallNeuro = 3
#> ... 1097 events were removed due to event being prior to first radiation or no radiation events
#> ... 1 patient(s) were removed
#> ... keep event agents:
#> KPS
#> measurement
#> mentalStatus
#> neuroFunc
#> overallNeuro
#> Radiation
#> Surgery
#> ... 194 events were removed due to no events in agents of interest
#> ... 0 patients were removed due to no events in agents of interest
#> ...these iois were removed:integer(0)

cat('...',nrow(fake_data$events), " events left for SPM after cleaning", '\n')
#> ... 1646  events left for SPM after cleaning
```

5. **Prep data by collecting patient info for each event**

```
fake_data <- merge(fake_data$events, fake_data$person, by='iois', all.x=T)
colnames(fake_data)
#>  [1] "iois"         "agent"        "agent_detail" "startdate"
#>  [5] "enddate"      "eventID"      "endEventID"   "eventName"
#>  [9] "survival"     "status"

# prep for each event, since age does change
fake_data <- prepDemographics(fake_data, fake_demo)
colnames(fake_data)
#>  [1] "iois"         "agent"        "agent_detail" "startdate"
#>  [5] "enddate"      "eventID"      "endEventID"   "eventName"
#>  [9] "survival"     "status"       "DOB"          "gender"
#> [13] "ethnicity"    "MSP"          "IDH1"         "ageDecade"
#> [17] "age"

# get survival labels, these also change
fake_data <- prepSurvivalLabels(fake_data)
colnames(fake_data)
#>  [1] "iois"          "agent"         "agent_detail"  "startdate"
#>  [5] "enddate"       "eventID"       "endEventID"    "eventName"
#>  [9] "survival"      "status"        "DOB"           "gender"
#> [13] "ethnicity"     "MSP"           "IDH1"          "ageDecade"
#> [17] "age"           "daysToDeath"   "survivalIn60"  "survivalIn180"
#> [21] "survivalIn270" "survivalIn360"

# get first tumor location
fake_data <- getTumorLocation(fake_data, fake_tumorInfo)
#> parsing tumor locations...
#> ...patients without tumor location :
colnames(fake_data)
#>  [1] "iois"                     "agent"
#>  [3] "agent_detail"             "startdate"
#>  [5] "enddate"                  "eventID"
#>  [7] "endEventID"               "eventName"
#>  [9] "survival"                 "status"
#> [11] "DOB"                      "gender"
#> [13] "ethnicity"                "MSP"
#> [15] "IDH1"                      "ageDecade"
#> [17] "age"                       "daysToDeath"
#> [19] "survivalIn60"              "survivalIn180"
#> [21] "survivalIn270"             "survivalIn360"
#> [23] "tumorLaterality"           "location: frontal"
#> [25] "location: temporal"        "location: occipital"
#> [27] "location: parietal"        "location: cerebellar"
#> [29] "location: thalamic"        "location: sellar"
#> [31] "location: corpus callosum" "location: pineal"
#> [33] "location: midbrain"
```

6. **Run spm**

   If you want to save the patterns and use in logit modeling:

```
runSPM(event = fake_data,
       suppList = suppList,
       maxgap = maxgap,
       maxlen = maxlen,
       maxsize = maxsize,
       tType = tType,
       outputDir = spmDir)
#>
#>
#>  Using spm patterns with rateChange ...
#> ...using file that already exists:  ~/gbm_spm_example/spm/transactions_rateChange.txt
#>
#>
#>  Reading from transactions file:   ~/gbm_spm_example/spm/transactions_rateChange.txt
#>
#>  Performing SPM...
#>
#>
#> parameter specification:
#> support : 0.4
#> maxsize :   2
#> maxlen  :   2
#> maxgap  :  60
#>
#> algorithmic control:
#> bfstype  : FALSE
#> verbose  :  TRUE
#> summary  : FALSE
#> tidLists :  TRUE
#>
#> preprocessing ... 1 partition(s), 0.03 MB [0.012s]
#> mining transactions ... 0 MB [0.008s]
#> reading sequences ... [0.018s]
#>
#> total elapsed time: 0.038s
#>
#>
#>  Getting spm events data ready for logit...
#> ...creating pattern and feature files in:  ~/gbm_spm_example/spm/sup0.4g60l2z2
#>
```

```
#>
#>  Searching for sequence patterns in events...
#>
#> 10 ... 0.04194951
#> ... finished search...
#> Time difference of 0.04751611 secs
#> ... testing...
#> ... number of unmatched frequent sequences support FROM TIDLIST (my results vs. arulesSequences,
#> ... setting NA values to 0 -- will return warning if non-spm features are NA beforehand
#>
#>  ...subset data based on different support thresholds...
```

otherwise:

```
# spm
pSPM <- getSeqPatterns(event = fake_data,
                       transFilename = file.path(outputDir, 'example_transactions.txt'),
                       createT = T,
                       support = minSupp,
                       maxgap = maxgap,
                       maxlen = maxlen,
                       maxsize = maxsize)
#>
#>
#>  Creating transactions file...
#>
#> ... file written to:  ~/gbm_spm_example/example_transactions.txt
#>
#>  Performing SPM...
#>
#>
#> parameter specification:
#> support : 0.4
#> maxsize :   2
#> maxlen  :   2
#> maxgap  :  60
#>
#> algorithmic control:
#> bfstype  : FALSE
#> verbose  :  TRUE
#> summary  : FALSE
#> tidLists :  TRUE
#>
#> preprocessing ... 1 partition(s), 0.03 MB [0.009s]
#> mining transactions ... 0 MB [0.007s]
#> reading sequences ... [0.011s]
#>
#> total elapsed time: 0.027s
pSPM$patterns <- as(pSPM$freqseq, "data.frame")
pSPM$patterns$sequence <- as.character(pSPM$patterns$sequence)

# days when pattern occur
patternDays <- findPatternDays(pSPM$patterns, pSPM$data, maxgap=maxgap)
#>
```

```
#>
#>  Searching for sequence patterns in events...
#>
#> 10 ... 0.0478518
#> ... finished search...
#> Time difference of 0.0533464 secs
#> ... testing...
#> ... number of unmatched frequent sequences support FROM TIDLIST (my results vs. arulesSequences,

# feature vectors to supply to logits
feat_vecs <- getFeatureVectors(patternDays, events=pSPM$data)
#>
#> ... setting NA values to 0 -- will return warning if non-spm features are NA beforehand
```